Computer Updates

The Newsletter For TS Computer Owners

Volume 3, Number 1 Winter 1986

THE NEW UPDATES

This issue marks a dramatic change in the newsletter. In our first two years, subject matter was limited rather strictly to applications and enhancements for the TS1000 version of ZX Pro/File. Now, the newsletter has broadened its scope. Instead of detailing just one program, the new Computer Updates will provide indepth information, new uses, program listings, hardware projects, computer theory, and a sprinkling of philosophical bull about TS computers in general.

If possible, I will try to confine my wanderings to the area of the products I sell. In this way, we both can benefit: I get a captive audience for my hyped-up advertising campaigns, and you can make more intelligent (?) decisions about the utility (???) of my products. Computer Updates will also function as an "extended documentation" for the programs and peripherals I sell. As new ideas, tips, discoveries, and uses come to being, you can read all about them in Updates.

THE GREAT COMPUTER DEPRESSION
(What is This World Coming To Dept.)

A few weeks back, I took out a subscription to INFOWORLD, a weekly newspaper featuring all the latest jive from the computer world. I figured that since I was "in the business", it would behoove me to "keep my ear to the rail". The paper would give me clues to what was happening in the other half of the computer industry.

Let me tell you, every time I read that paper I get depressed! Its not that INFOWORLD is a bad paper. It is the news itself that is so depressing. For instance, did you know that Apple computer has filed suit against their founding father, Steve Jobs. Apple alleges that Jobs has stolen trade secrets!

Another industry giant, Microsoft, has been accused of lacing their software with a rather nasty message intended to scare the bejeezers out of anyone who finds it. Apparently, some owners were trying to break into and modify the program when a message came up stating that they had no business breaking into the program and that the machine was now destroying all data on the disk. Microsoft really knows how to win friends and inspire confidence!

The more I learn about the rest of the computer world, the more I appreciate ZX/TS world. Our dealings are with PEOPLE-- individual mom and pop type businesses. The monsters who stab the knife in the backs of both their customers and their founders ignore us. We're too small potatoes for them. And that's a good thing.

But that's enough of the hard cruel world. Let's retreat to the Timex tranquility. This issue has some very interesting projects and tips. Read on, read on.

ANOTHER WAY TO SAVE DATA IN PRO/FILE 2068

When you're running Pro/File 2068 and you type "SAVE" from the main menu, program line 107 is executed which performs a normal program save of the Basic program and all the variables including D$, the large character array which contains all your data. Many people have asked if it would be possible, instead, to save just the data and not the program lines.

There are two advantages to saving in this way. First, since you would not need to save the lines of the program, cassette times would be shortened. Actual time would vary depending on how many records you have added. Second, the data you save could be loaded into other programs--either other versions of Pro/File 2068 or completely different software.

There have been quite a few people who made some modifications to Pro/File 2068 only to discover that when they tried to Load their data into their new version, they also loaded in the old unmodified program lines. The only solution was to retype all the data back into the enhanced Pro/File.

By changing the Save/Load procedures, you won't need to re-enter all that data. Saving time will be shorter, and you will be able to isolate the data from the program. This means you can write other Basic programs to operate on your files in a way that Pro/File does not (such as a search and replace procedure).

Computer Updates is published quarterly in the winter, spring, summer and fall. Subscription price is $12.95 per year. In Canada and Mexico please add $3 postage. All other foreign subscriptions please add $9 postage. Back issues are available as single volume, 4 issue sets. Volume 1 is $9.95, volume 2 is $9.95

Edited and published by:
Thomas B. Woods
P.O. Box 64, Jefferson, NH 03583
(603) 586-7734

This technique converts the old program save and load into a CODE save and load. Alter the Basic lines 107 and 5510 so they read as follows. You will need to do this in every copy of Pro/File you use. After the lines are changed, use the GO TO 1 command to get the program running again. Now, with a spare cassette in your recorder, type "SAVE" from the main menu to save just your files.

```
 107 IF x$="SAVE" THEN LET d$( T
O 5)=STR$ p: SAVE f$CODE 6+PEEK
23627+256*PEEK 23628,p: LET d$(
TO 5)="*SEAR": GO TO 1

5510 PRINT AT 15,1; PAPER 5; INK
 0;"WHAT FILE NAME DO YOU WISH
  ";AT 17,1;"TO LOAD ";: INPUT F
$: PRINT f$: LOAD f$CODE 6+PEEK
23627+256*PEEK 23628: LET p=VAL
d$( TO 5): LET d$( TO 5)="*SEAR"
: GO TO 1
```

After each Pro/File you have is saved in this way, follow the instructions on page 99 of the big book to make a back-up copy of your original master. Once step 5 is completed, alter lines 107 and 5510 just as you did above. Then finish executing steps 6 through 9 of the back-up making procedure.

When you're finished, your new version will operate in exactly the same way as before except saving and loading times will be reduced dramatically.

What do lines 107 and 5510 do? You can read in the 2068 operator's manual that the SAVE "name"CODE command, sends raw bytes of memory out to the tape recorder. You must tell the machine the address of the first byte and the number of bytes to send. Since D$ is always the first variable stored in memory, it is easy to calculate its beginning address by peeking the VARS system variable (23627 and 23628) and adding 6 to it. The Basic variable, P, always tells how many bytes of D$ actually hold data. These two items provide the necessary information so the computer will know what and how much data is to be saved.

Because each save varies in length, Pro/File must update the variable P after a load so its new value corresponds with the newly loaded data. That's why the first part of line 107 says,

LET D$(TO 5)=STR$ P

Normally the first five characters of D$ will be the beginning of the *SEARCH IS COMPLETE file. Line 107 temporarily overwrites this with the number P. After the save is finished, the same line restores D$(1 to 5) to its old self.

Therefore, when line 5510 is called upon to load data, the computer looks at the first 5 characters to get the corresponding value for P. Before line finishes, it too, restores these 5 characters to "*SEAR" so Pro/File will function correctly.

Because data is saved as CODE (as opposed to DATA D$), you must load the files into other programs as CODE as well. On the surface, it would seem that this is less than ideal because most often you'd like to load it as a string. Actually, this approach is more versatile. In addition to being able to load the data as CODE, you can also do just what Pro/File does--create an array first, then fill it up with the CODE. Essentially, all you have to do is work up a program line that looks something like this:

```
CLEAR: DIM D$(28000):LOAD " "CODE 6
+PEEK 23627+256*PEEK 23628
```

The CLEAR command insures that D$ which is created by the next statement will be the first variable in memory and therefore at the address calculated by peeking VARS.

In the dimension statement above, there is nothing that prevents you from making D$ any size you wish as long as you don't try to make it smaller than the value for P. In other words D$ can be anything larger than P. If P equals 2000, you could Dim D$ to any size greater than 2000. But don't try to make it smaller. That's like trying to stuff 10 pounds of mud in a 5 pound sack. It just won't work. Re-dimming D$ to a smaller size is very useful at times, but means that whatever data is stored in the array will be lost. This trick lets you save off your files, put them in a smaller array without having to retype everything back in by hand.

HERE'S A CHEAP HOME BREW TONE DECODER THAT WORKS GREAT!!

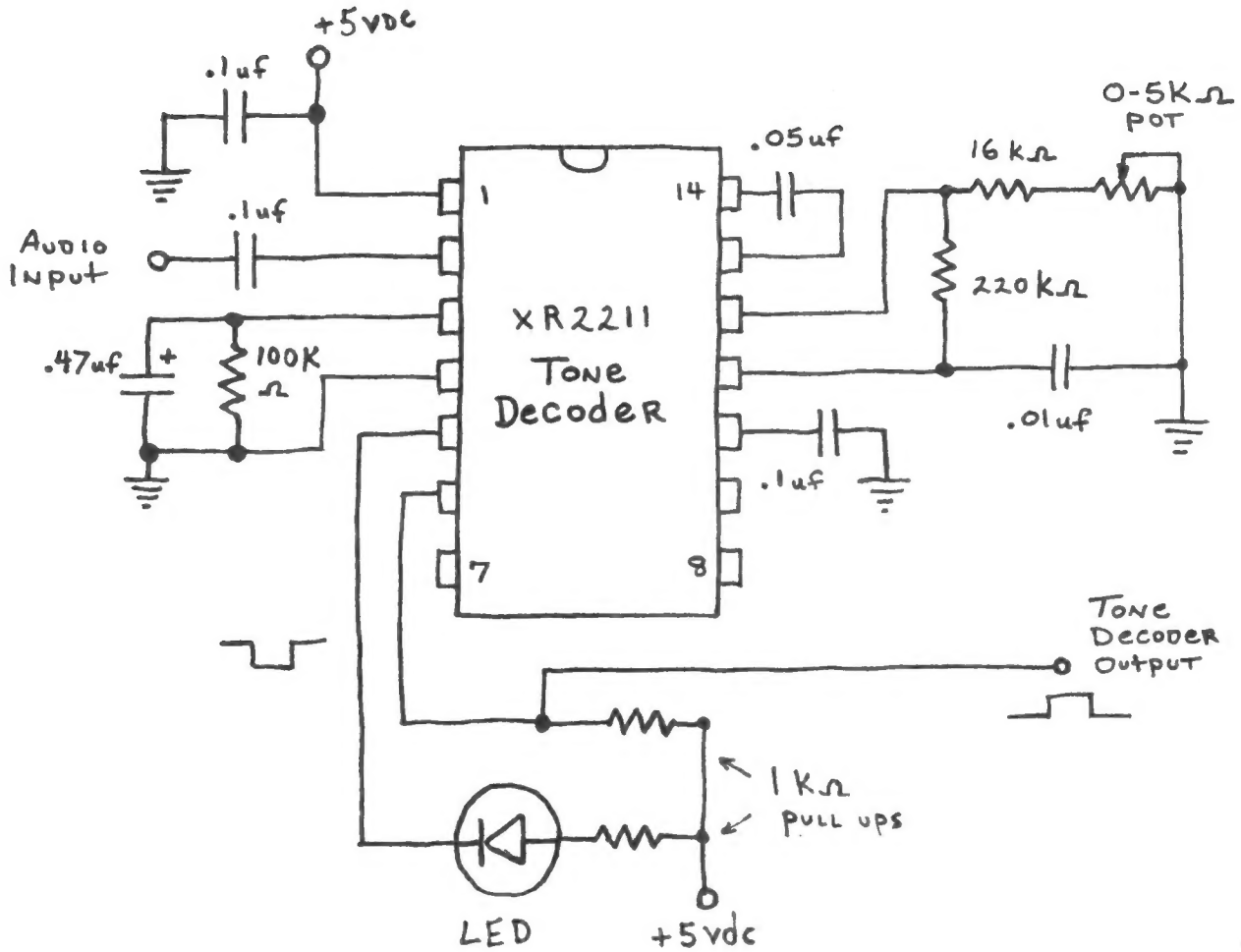
What's a tone decoder, you ask? Well, a tone decoder is a vital piece of equipment which must be hooked up in the Morse Code Translator program for the Experimenter's Universal In/Out Port. It is a circuit that listens to audio dots and dashes from the radio speaker and whenever the sound falls within a narrow selected frequency, an output goes high. Otherwise, the output is low. This output signal is the one that is read by the computer when the program converts the dots and dashes into letters of the alphabet.

You can buy tone decoders from a variety of sources, but they tend to be rather expensive: $75 to $375 depending on quality and features. An alternative to buying a ready to go unit is to build your own decoder using the diagram shown here. For $5-$10 you can buy all the parts you need, and it will perform superbly. In fact, this circuit works far better than the commercial rig I purchased for $100. Building it is a pleasant evening's project. Considering its simplicity, its just amazing how well it works. Even in noisy, interference filled, static conditions, this decoder pulls out faint barely audible morse code. It literally finds the needle in the haystack.

The parts can be purchased at Radio Shack or your favorite electronics supply house. I built my decoder on one of those experimenter's solderless breadboards. This allowed me to simply plug the components into the board without soldering them. Usually, soldering all connections is recommended by the engineers. I won't argue with that, but I really don't see how soldering could improve the operation of this decoder other than to make it more permanent. Perhaps the best approach would be to build the decoder up on the breadboard until you're happy with the way it works. Then for the final version, go back and solder the components together permanently on a printed circuit board.

Here is a list of parts you will need:

1-Experimenter breadboard
1-XR2211 Tone Decoder Chip
1-LED (any type)



Capacitors

3-.1 uf capacitors
1-.47 uf electrolytic capacitor
1-.05 uf capacitor
1-.01 uf capacitor

Resistors (all 1/4 watt)

1-0 to 5 Kohm variable resistor
2-1 Kohm
1-100 Kohm resistor
1-16 Kohm
1-220 Kohm

As you build the tone decoder, try to keep the component leads as short as possible. You don't need use the values shown exactly. The closest size you have will work just as well. The exceptions to this rule are the .47 uf electrolytic capacitor and the 100 Kohm resistor. For these use only the values shown. By increasing the value of the 220 Kohm resistor connected between pins 11 and 12 of the chip, you will narrow the "window" of the tone frequencies which will send the output high. As shown, the range is about 250 hz. Increase this resistor to a megohm and the window will be only about 20 hz wide.

This makes for a VERY selective tone decoder, but one that is harder to tune your radio to. The variable resistor connected between pin 12 and 4 (ground) lets you move the window up or down in frequency.

To use this tone decoder with the Morse Translator, apply 5 volts to the supply pin (1) and connect pin 4 to ground. You can use the computer's power supply if you wish, but this will sometimes allow an objectionable amount of noise to pass into the radio. A better approach is to use a separate 5v supply.

Use shielded cable to connect the speaker output of the radio to the tone decoder input. Hook the decoder output (pin 6) up to bit zero of the input port following the recommendations made in the Translator documentation. Turn on your radio and tune into some morse code. When you find a transmission, adjust the tuning of both the 0 to 5 Kohm variable resistor, and also the radio. When the tone is right, the decoder locks into it, and the LED will blink in time to the dits and dahs.

USE "DO" FILES TO CHANGE PRINTER FORMAT IN ZX PRO/FILE

If you find that you frequently need to change the print format (DEFP) in ZX Pro/File for the TS1000, you can save a lot of combersome keypresses by assigning a DO file to do it for you. Since the print format is stored in the three basic variables, C1, C2, and S, it is a fairly simple matter to set their values in a program line rather than input them from the keyboard.

Suppose you want to print out 3 lines starting at line 6 of a file. Your DO file program lines could look something like this:

```
7000 LET C1=8
7010 LET C2=6
7030 LET S=3
7040 GOTO 17
```

Remember, C1 always equals 2 more than the number for the first line to lprint. By setting C1 to 8, line 6 will be the first to be lprinted. After you add these lines, set up a DO file which jumps to the location of your instructions. In this case it is line 7000, but it could be anywhere there is space. When you enter the DO file as a Search Command, the lines will be executed, setting the print format as desired. Then, the last line jumps you back to the Main Menu. NEAT!

Can anyone think of a way to make a DO file set the print format to just one line and then print lines in an order different from the way they appear on the screen? This would be like printing line 6, then 2, then 3, then 8.

UNCLASSIFIED

Sell that piece of gadgetry that failed the smoke test; or that extra memory or printer. Non-commercial ads just $5 for 5 lines!

FOR SALE: Two TS1000's, one TS2040 printer, Byte-Back 64K RAM, EZ-Key 80 keyboard, Q-Save filter and program, and miscellaneous books, magazines, and programs. All for $275 or make an offer on pieces. Call (915) 944-7970 after 6pm.

FOR SALE: Dot-Matrix Printer--C. Itoh Model 7500 Brand new, still in carton. Centronics parallel type, print looks just like the Prowriter. $250 Call Tom Woods, (603)586-7734

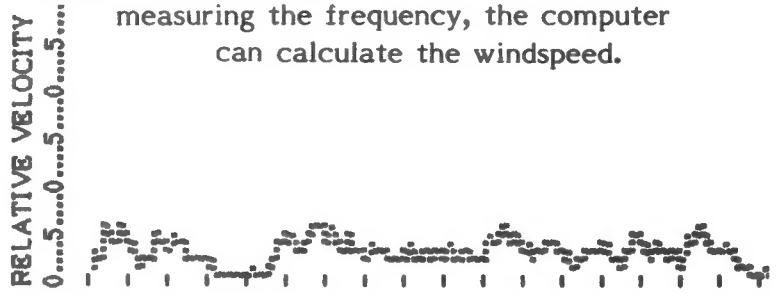
BUILD A COMPUTERIZED ANEMOMETER USING THE EXPERIMENTER'S I/O PORT

This project is the first in a series of articles which will chronicle my ongoing experiments in the development of a low-cost completely computerized weather station making use of TS computers, the Experimenter's Universal Input/Output Port, and a collection of common household items.

I think you will enjoy following my trials and tribulations. After completing this first part, it is apparent that there shall be some very interesting computer concepts covered here which will be applicable in many different situations. If you decide that you'd like to try building some of the projects yourself, welcome aboard. Please tell us about your ideas, experiments, findings, failures and successes. Bear in mind that above all else, I am a tinkerer; not an engineer or electronics wizard. So please, all you engineers and electronics wizards out there, be forgiving when you see my designs. Help out if you can by sharing your thoughts.

An anemometer is a device which tells you how fast the wind is blowing. Usually, anemometers work by having some sort of arm stuck up in the wind. The wind spins the arm which is connected to the shaft of a generator. The harder the wind blows, the faster the arm spins. The generator will produce higher voltage. By measuring the volts, you can calculate the windspeed.

The anemometer I built operates on roughly the same principal, except instead of generating a voltage, the spinning arms of the rotor slice through a light beam. The computer is interfaced to the anemometer through a light detector so what the computer sees is a series of high and low pulses as the arms break the light beam. As windspeed increases, and the anemometer spins faster, it is not a higher voltage which is produced but a higher rate of pulses per unit of time. In other words: a higher FREQUENCY. By measuring the frequency, the computer can calculate the windspeed.



Construction of the Mechanical Parts

Look at figure 1 to get an idea of how the anemometer is constructed. A ping pong ball which has been cut in half is used as the wind collectors. When you cut the ping pong balls, use a SHARP utility knife. After the initial cut, trim the halves with scissors until each is exactly the same size.

Once you're finished with the ping pong balls, set them aside and build the shaft assembly. First, take a small mending plate (or similar metal bar 3 inches long by 1/2 inch wide) and mark its center point. Cut a piece of 1/4 inch copper tubing about an inch and a half long. Ream any burrs that formed on the inside of the tube as a result of the cutting process. Make sure that the ends of

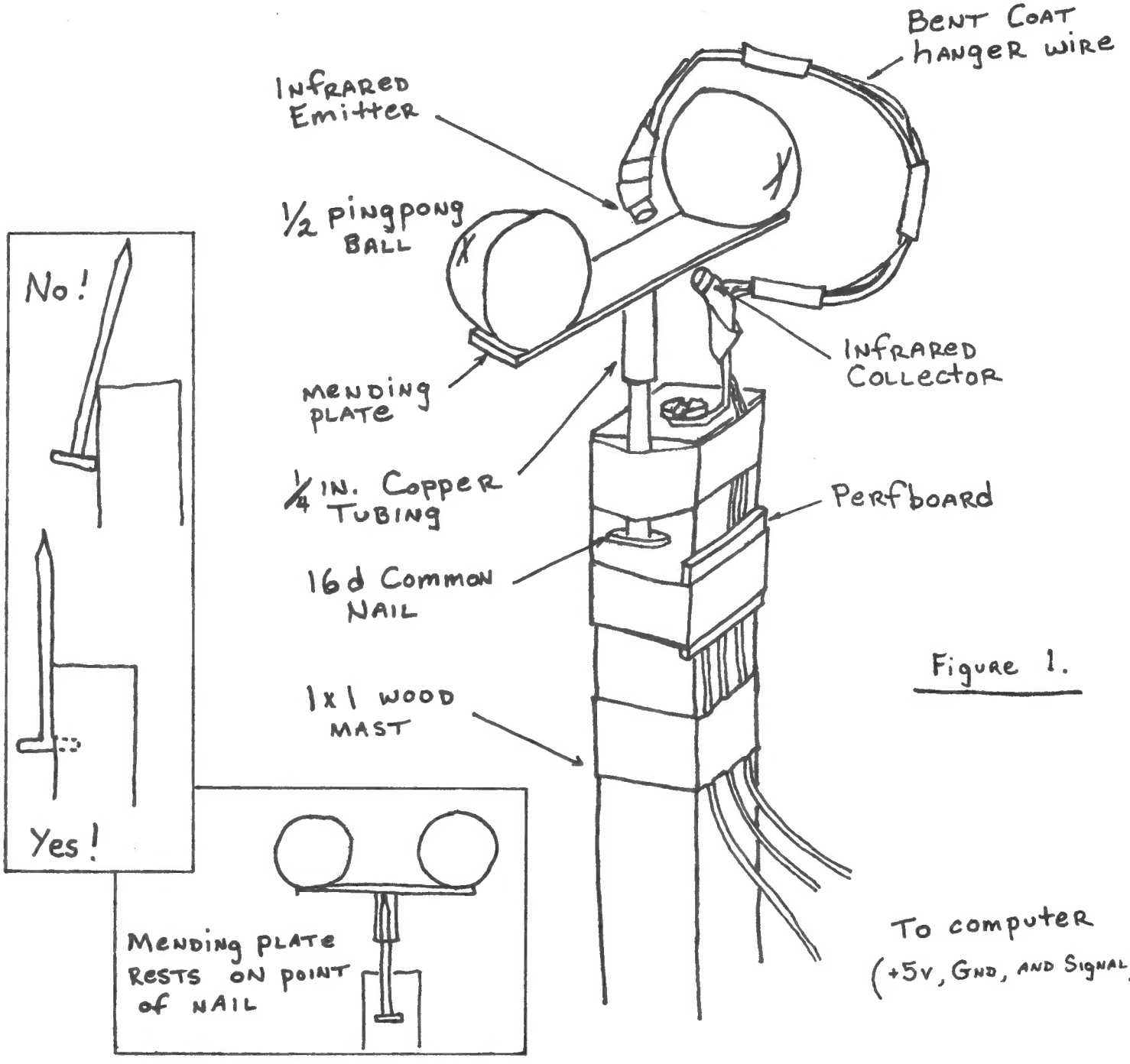


Figure 1.

the tube are straight and square. Run the ends across a grind stone or file them if necessary. A 16 penny common nail should freely pass through the tube. Now use super glue to stick one end of the tubing to the mending plate. You should be able to look inside the tube and see the center mark of the mending plate in the exact center of the tube. Go slow and get it right!

Use vice grips or pliers to squeeze the head of the 16 penny nail into the end of a wooden mast. This will allow the nail to stick up straight. Wrap electricians tape tightly around the nail and mast to secure.

Place the tube and mending plate rotor over the nail. The rotor should spin with very little drag. Mix up some epoxy and glue the ping pong ball halves onto the ends of the mending plate. Don't forget to face them in the right direction, and take pains to insure that the cups are spaced equally from the center. While you're at it, gob a generous amount of epoxy in the area where the tubing connects to the mending plate. Spread the epoxy right around the tube and over the top face of the mending plate. A lot of stress will build up here in a high wind.

As in all rotating objects it is important to try to keep the rotor properly balanced. That is why you should be careful in your positioning of the tubing and the ping pong balls. Before you clean up the epoxy, try spinning the rotor gently. If you see it wobble and shimmy, place a bit more epoxy on the mending plate to bring it into balance.

Electronics Construction

While you're waiting for the epoxy to harden, begin work on the electronic portions of the anemometer. As mentioned previously, a light emitter and detector circuit must be located at the rotor. It must be positioned so that the light path between the two will be broken by the mending plate as it spins. Use the schematic diagram and the drawing shown in figure 2 as a guide in fabricating the circuit on a small piece of perfboard. Use about 6 inches of lead wire soldered to both the infrared emitter and detector.

After all connections are soldered, cover the board completely with electrician's tape to protect it from the weather. Then tape it to the wooden mast as shown in figure 1.

Cut up a coat hanger and form it in a shape similar to that shown in figure 1. Use a wood screw to fasten it to the end of the mast. You want to bend the coat hanger so that it will hold the emitter and detector about an inch apart, and at the same time allow the cups of the rotor to spin freely. Use more electrician's tape to secure the emitter and detector as well as the lead wires to the coat hanger wire. Be sure the emitter is lined up so its light can be seen by the detector.

If you wired the circuit up properly, you should have three leads coming down off the mast. They go into your computer room. The +5vdc and ground lines can connect directly to your computer. I did this by soldering two wires to the power pins on the underside of the I/O port edge connector. If you turn the port upside down, the +5 volt line is on the extreme right side. The ground is the fat trace immediately to the left of the key slot. The third line coming from the mast is the data line which must be fed to the input of an LM339 quad comparator chip which is shown in figure 3.

The comparator chip functions as a "level detector". It compares the voltage coming from the light detector (at pin 5) with a reference voltage (at pin 4) which can be adjusted by turning the 100 Kohm variable resistor shown in the schematic. Whenever the voltage from the light detector is greater than the reference voltage, a high signal is output to pin 2. When the voltage from the detector is less than the reference, pin 2 goes low.

The comparator is essential for cleaning up spikes and glitches coming in from the light detector. The adjustable reference voltage takes into account different voltage drops which occur when you set the anemometer up at various distances. The comparator's output is a nice sharp 0 to 5 volts--just perfect for sending into the computer via the input port.

Build the comparator circuit on a piece of perfboard or a solderless breadboard. Supply it with 5 volts from the computer. Every data sheet on the LM339 I've read says that you must ground all pins of the chip which are not used. While this may very well be a good practice, I have found that doing this has no effect on the circuit. So for convenience and simplification, don't bother grounding the pins. Besides, we may want to use the 3 unused comparators in the LM339 for something else later.

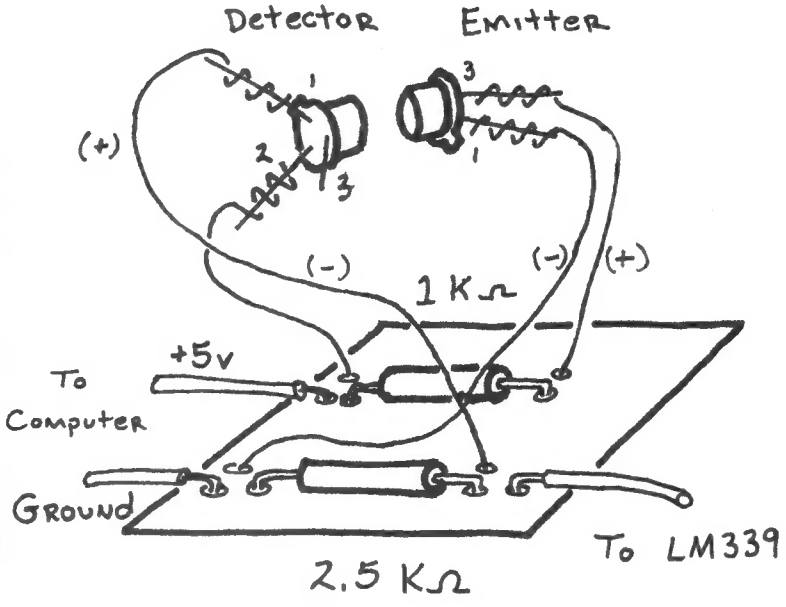
Testing the Anemometer

Before you permanently install the anemometer, you will want to test it inside. With my rig, I used a four foot long mast. It was held in a vice on the floor. A fan was directed at the cups on the mending plate. The fan gave it a nice slow spin of about 1 revolution per second. In your tests, you will find that it is much easier to see what's going on if you keep the anemometer from spinning too fast. For now, the slower it spins, the better.

Hook up wires to the +5 volt and ground lines of the port board as previously described.

Plug the port into the back of the computer and turn it on. For testing, you don't need the computer for anything other than its power supply.

Temporarily connect the power lines up to their respective leads of the comparator circuit and the emitter/detector circuit. Also, tie the data line from the anemometer to its proper location in the LM339 circuit. Use wire nuts or just twist the wires



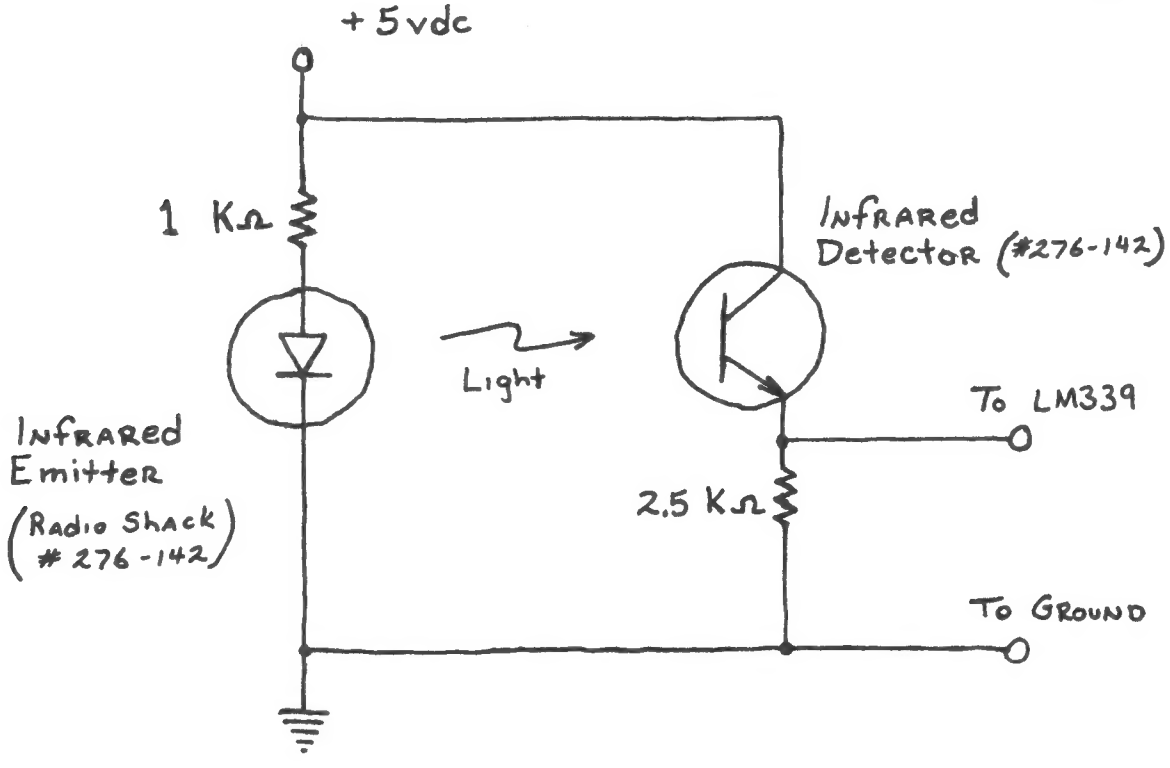


Figure 2.

together. Turn on the fan so it spins the anemometer. Use a volt meter to measure the voltage between pin 5 of the comparator and the ground.

This is the data line coming in from the light detector. The actual voltage measured depends on many factors. It will be different for each set up. What you should see is a definite swing of at least 1 volt as the anemometer spins. In my test, I measured a high voltage of just under 3 volts and a low voltage of just over 1 volt.

Now measure the reference voltage at pin 4 of the LM339. By turning the variable resistor in the circuit, you should be able to adjust this voltage through the range of 0 to 5 volts. Set the voltage at the pin to the highest possible setting (5 volts).

If you measure the output of the LM339 (pin 2), you will see that it is now very close to 0 volts. This is because the input from the light detecter is less than the reference voltage (5 volts at pin 4). Watch the voltage at pin 2 while you slowly turn the variable resister to lower the reference voltage. At some point, you will begin to see the pin 2's output voltage swing between 0 and 5 volts as the anemometer spins. It is at this point where comparator is allowing the data from the light detector to pass through and you know you connected the circuit properly.

One last thing you should do before you install the anemometer outside is to lift the rotor off the nail and paint the bottom side of the mending plate black. I discovered that if the mending plate was not blackened, bright sun can reflect off the rotor arm and trip the infrared detector. This was particularly noticable (and also a real headache til I discovered what was happening) on a bright day with fresh snow cover.

Outdoor Installation

Windspeed, of course, varies tremendously from place to place. It can show significant differences between ground level and 10 feet up in the air. Bushes, trees, buildings, and hills all cause variations which are quite unpredictable. You want to locate the "most average" spot you can for your anemometer. You also want to keep it as close to your computer as possible.

Keep the anemometer well away from (or above) trees, buildings, or other physical obstructions which can cause sheltering or turbulence. Many people mount their anemometers on rooftops. If this is not possible, construct as tall a mast as you can (like a 16 foot long piece of 2x4) and place it as best you can "out in the open".

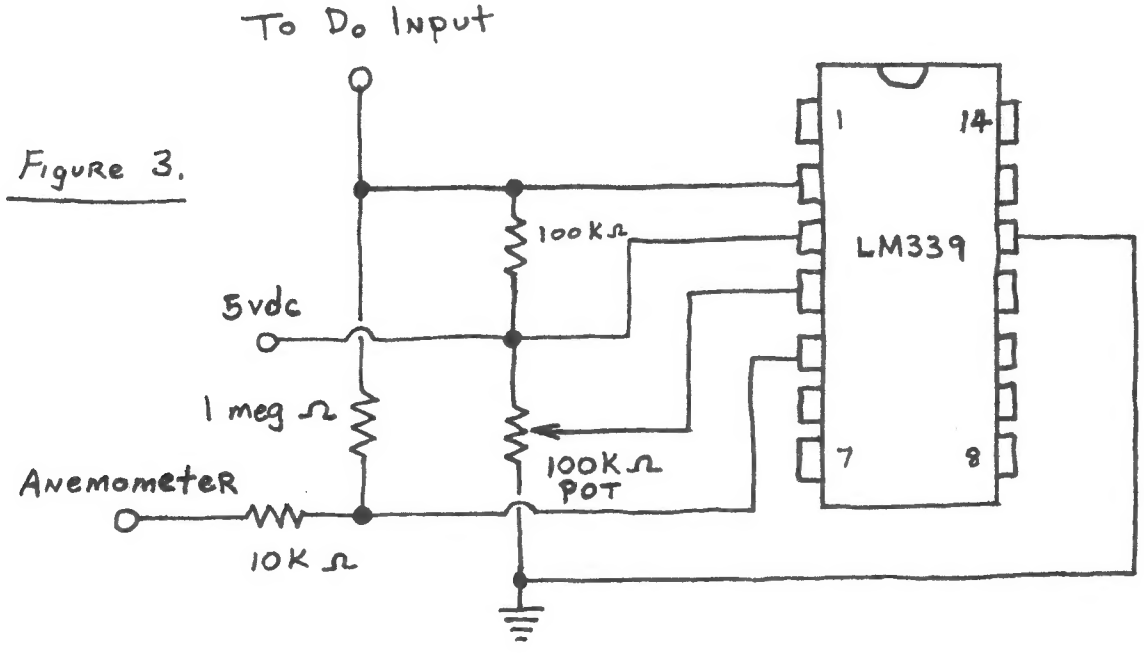


Figure 3.

Interfacing to the Computer

With the computer turned OFF, plug your printer interface and the Universal In/Out Board into the back of the computer. You may need to fabricate some sort of extender off the back of the machine to accomplish this. If you have not done so already, tap into the +5v and ground lines as described previously and use them to power the LM339/light detector circuits. Hitch the output of the LM339 (pin 2) to bit 0 of the input port. Your computer is now ready to measure the wind.

Now the Software

Enter this listing into your TS1000 and save it. (2068 OWNERS NOTE: a 2068 version of this program appears at the end of this article).

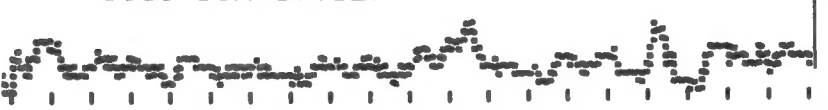
```
  10 LET A$=CHR$ 1+CHR$ 0+CHR$ 0
+CHR$ 118+CHR$ 62+CHR$ 60+CHR$ 5
0+CHR$ 52+CHR$ 64+CHR$ 219+CHR$
223+CHR$ 95+CHR$ 58+CHR$ 52+CHR$
 64+CHR$ 254+CHR$ 0+CHR$ 200+CHR
$ 219+CHR$ 223+CHR$ 187+CHR$ 40+
CHR$ 245+CHR$ 3+CHR$ 24+CHR$ 241
  20 LET WIND=3+PEEK 16400+256*P
EEK 16401
  30 SLOW
  40 PRINT "RELATIVE VELOCITY:";
AT 5,0;"TRANSITIONS PER SECOND:"
  50 PRINT AT 3,0;"0....1....2..
..3....4....5....6"
  60 LET Y=0
  70 LET P=90
  80 PRINT AT 16,0;"PRESS -P- FO
R LPRINT                -Q- FO
R SCREEN ONLY"
  90 LET X=USR WIND
 100 UNPLOT Y,38
 110 PLOT X,38
 120 LET Y=X
 130 PRINT AT 5,23;X;"   "
 140 IF INKEY$="P" THEN LET P=10
00
 150 IF INKEY$="Q" THEN LET P=90
 160 GOTO P
1000 REM HARD COPY
1010 LPRINT CHR$ 155;"RP";"RELAT
IVE VELOCITY"
1020 LPRINT "  0....1....2....3.
...4....5....6"
1030 DIM S$(32)
1040 LET T=0
1050 LET P$="..................
............."
1060 LET Z=1
1070 LPRINT CHR$ 155;"RT02"
1075 LET X=X+1
1080 IF X<=Z THEN LET D$=S$( TO
X-1)+P$(X TO Z)
1090 IF X>Z THEN LET D$=S$( TO Z
-1)+P$(Z TO X)
1100 IF T=0 THEN LPRINT "-";D$
1110 IF T<>0 THEN LPRINT "  ";D$
1120 LET T=T+1
1130 IF T=10 THEN LET T=0
1140 LET Z=X
1150 LET P=1075
1160 GOTO 90
```

When you RUN the Basic (Don't use GOTO here), the computer plots the current relative wind speed on the screen in the form of a "sliding pixel". The program will also Lprint a graph of the data if you press "P". Because the Basic spends most of its time reading the anemometer, it checks the keyboard only once per second or so. If you find that pressing "P" has no effect, just hold your finger down a little longer.

You will need to alter the Lprint routine beginning at line 1000 to match your individual printer. The listing works with the Prowriter and the Memotech centronics interface. In line 1010, the command, LPRINT CHR$ 155;"RP" puts the Prowriter into proportional spacing mode. The LPRINT CHR$ 155;"RT02" in line 1070 changes the line feed pitch to just 2 dots per line feed. The result is a fairly nice graphics printout just by printing a series of periods (.)! A sample printout of a fairly calm day here in Jefferson is shown.

To explain how this program works, let's review what the anemometer is capable of doing so we can see what tasks the software has to perform. Esentially, we have a rather fancy switch mounted at the top of a long stick. The rotor and wind cups cause the switch to open and close as the wind blows. The faster the wind blows, the more frequently the switch opens and closes.

To determine the wind speed, therefore, we must find out just how often the switch opens and closes in a fixed unit of time. One way to measure time with the TS1000 is to



capitalize on the hardware generated interrupt signals which occur at the rate of 60 pulses per second.

When the computer is in the SLOW mode, each interrupt causes the system to generate the TV display. In addition, the system variable FRAMES (address 4034 and 4035 hex) is used to count each interrupt. The value stored in FRAMES decrements each time an interrupt signal is generated by the computer.

A short machine code routine counts the number of times our light detector switch opens and closes during the period of 60 interrupts (approximately 1 second). The program samples the input from the light detector and compares it with the previous sample. If the two are different, meaning the rotor arm has either left or entered the light path, a counter increments. Each transition from low to high and high to low adds one to the count. This process continues until 60 interrupts have occured. Then the machine code returns to Basic.

The rotor causes 4 transitions in the light detector per revolution. In one complete turn, one end of the rotor breaks the light path causing a change from high to low. As the arm continues out of the path, the second transition from low to high occurs. The revolution completes when the other end of the rotor causes two more transitions from high to low and low to high.

The actual machine code that does the counting appears below. Although it is written for use on the TS1000 type computer, you could very easily run it on the TS2068. Only the address of FRAMES (4034 hex) is different. The necessary changes are incorporated in the 2068 listing.

```
010000    BGIN LD BC,0000
76             HALT
3E3C           LD A,3C
323440         LD (4034),A
DBDF           IN A,(DF)
5F        STOR LD E,A
3A3440    CNT_ LD A,(4034)
FE00           CP 00
C8             RET Z
DBDF      SMPL IN A,(DF)
BB             CP E
28F5           JR Z,CNT_
03             INC BC
18F1           JR STOR
```

To start, the code loads BC with the initial count of zero. Then the HALT instruction causes the computer to wait until the first interrupt. The hex number 3C (or 60 in decimal) is placed in the FRAMES system variable (address 4034). This represents the time the machine code will spend counting transitions which the computer now starts to do. The IN A,(DF) instruction reads the initial state of the anemometer's light detector. It will either be high or low. The value found is stored in E. After each sample, FRAMES is checked for a zero value by the code starting at the label CNT_.

The interrupts cause FRAMES to decrement "automatically" so after each, FRAMES will equal FRAMES-1. Eventually, it will reach zero and the computer will return to basic. As long as FRAMES does not equal zero, the code labeled SMPL will execute. Here, another IN A,(DF) instruction samples the port. The value found is next compared with the value in the E register. As long as the two are the same, it means no change has occured, and the code jumps back to see if FRAMES has reached zero yet. When the values in E and A are different, we have a transition. Therefore, the code increments BC and loops back to the label STOR which puts the new value in E before repeating the whole sequence.

Because BC is used to count transitions, the machine code returns with X=BC or put differently, X equals the total number of transitions.

The Basic portion of the program performs the tasks of storing the machine code and executing it. Then it displays the information generated by the machine code graphically. I am sure you will want to experiment with different display formats. What is presented is intended to be a starting point--not the ending point.

The only lines you should NOT tamper with (unless you can program in machine language) are the first three: lines 10, 20, and 30. Line 10 contains the machine code. It is stored in the form of a string rather than the customary REM line because the second machine code instruction, HALT, cannot be placed in a REM without crashing the computer. Thus, we keep the code out of a REM line!

Line 20 determines the starting address of the code so we know where to jump to execute it. Since A$ is the first variable initialized, its location can be found very easily. All you have to do is add three to the value stored in VARS (address 16400 and 16401 decimal). This is what is done by line 20.

Line 30 is essential to the execution of the machine code. Whenever you run code which uses the HALT instruction, you MUST put the computer in SLOW first. If the computer executes such code in FAST mode, you crash because interrupts are disabled in FAST. Without interrupts, the HALT instruction will cause the computer to wait forever.

Aside from that, you are completely free to add whatever additional Basic you want. Just be sure you write your program so you can use RUN to start it. In this way, the machine code will always be in the right place and you will always be in SLOW mode. Any time you want the rotation rate of the anemometer just execute the command LET X=USR WIND. This makes X equal to the number of times the rotor sliced through the light beam in one second. From that, you can print it, plot it, store it, or whatever.

Converting "Relative Velocity" into MILES PER HOUR

If you have been wondering why I have not programmed this into the Basic already, well, errrr, it's because I haven't figured out how to do it yet. Surely, you helpful readers can come to my rescue. I have heard of others who devised an ingenious plan of sticking the anemometer out of their car window and driving down the road at several known speeds. This would work in our case too, but hooking up the computer, a monitor, etc. would be quite an undertaking.

Another method which is less exact, but more appealing (to me at least) is to call several local airports, weather stations, etc. and take an average which could be factored in to the program. By calling several such places, and doing it many times, you should be able to reach a fairly high degree of accuracy.

Perhaps easier still is the "estimating by phenomena" approach which I copped out of the book, WIND AND WINDSPINNERS by Michael A. Hackleman. Using this method, you translate trasitions per second into miles per hour by relating other natural phenomena. Refer to the table below which was taken from Hackleman's book:

MPH	Phenomena
0	Smoke rises vertically
1-3	Smoke drifts in direction of wind
4-7	Wind felt in face, leaves rustle
8-12	Leaves, small twigs in constant motion
12-18	Dust, loose paper rises, small branches move
19-24	Small trees begin to sway. Wavelets form on pools
25-31	Large branches move. Phone wires whistle
32-38	Whole trees move. Walking difficult
39-46	Twigs and branches break off trees
47-54	Flower pots and house tiles are removed

Conclusion

I hope this article perks your interest this unusual use for your computer. In future issues, I will incorporate other weather monitoring devices into the foundation layed here. Wind direction is certainly high on the list, as is temperature, barometric pressure, and humidity. Other data which would be fun to collect are sun light intensity, soil temperature, and precipitation (both quantity and acidity). A seismometer (earth tremors) would be interesting to experiment with as well. The exciting thing is that all these things can be done with TS computers and a little Rube Goldberg type inventiveness, and all of it requires the use of just one computer and one input port. Have fun!

TS2068 Listing

```
  10 LET A$=CHR$ 1+CHR$ 0+CHR$ 0
+CHR$ 118+CHR$ 62+CHR$ 195+CHR$
50+CHR$ 120+CHR$ 92+CHR$ 219+CHR
$ 223+CHR$ 95+CHR$ 58+CHR$ 120+C
HR$ 92+CHR$ 254+CHR$ 0+CHR$ 200+
CHR$ 219+CHR$ 223+CHR$ 187+CHR$
40+CHR$ 245+CHR$ 3+CHR$ 24+CHR$
241
  20 LET WIND=3+PEEK 23627+256*P
EEK 23628
  40 PRINT "RELATIVE VELOCITY:";
AT 5,0;"TRANSITIONS PER SECOND:"
  50 PRINT AT 3,0;"0....1....2..
..3....4....5....6"
  60 LET Y=0: LET P=90
  80 PRINT AT 16,0;"PRESS -P- FO
R LPRINT                -0- FO
R SCREEN ONLY"
```

```
  90 LET X=USR WIND
 100 PRINT AT 2,Y/2;" ";AT 2,X/2
;"■": LET Y=X: PRINT AT 5,23;X;"
 "
 140 IF INKEY$="P" THEN LET P=10
00
 150 IF INKEY$="Q" THEN LET P=90
 160 GO TO P
1000 REM HARDCOPY
1010 LPRINT CHR$ 27;"P";"RELATIV
E VELOCITY"
1020 LPRINT "  0....1....2....3.
...4....5....6"
1030 DIM S$(32): LET T=0: LET P$
=".............................
..": LET Z=1
1070 LPRINT CHR$ 27;"T02"
1075 LET X=X+1
1080 IF X<=Z THEN LET D$=S$( TO
X-1)+P$(X TO Z)
1090 IF X>Z THEN LET D$=S$( TO Z
-1)+P$(Z TO X)
1100 IF T=0 THEN LPRINT "-";D$
1110 IF T<>0 THEN LPRINT "  ";D$
1120 LET T=T+1: IF T=10 THEN LET
 T=0
1130 LET Z=X: LET P=1075: GO TO
90
```

BUGS, NEWS, AND MISCELLANEA

Pro/File 2068 owners take note! A few fustrated folks who added the new Machine Code Sort capability from the BREAKTHROUGH newsletter experienced crashing problems whenever they tried to alphabetize files by one particular line, but ordering by any other line worked fine. This is a bug in the machine code--not a typing error.

Robert C. Fischer, 221 Scoggins St., Summerville, GA 30747 was able to pinpoint the exact effect. He wrote, "If you compare a line of data in different files, and the data matches EXACTLY, but if in one case the data is the last line of a file while in the other it is not, the program crashes. The routine can't handle a comparison of * and code 1 which signify the end of file and end of line markers respectively."

What does this mean? Well, basically, it means that you should avoid sorting any file on the basis of the last line. A more prudent approach would be to use the first few lines of each file to store "sort codes" and use the remainder of the file to store other data which would not be sorted. Unfortunately, it would be a major task to fix this bug. It is far easier to arrange your data in such a way that the bug will never rear its ugly head.

To all of you who ran into problems with this and thought you were losing your sanity, please accept my apologies for not spotting this problem. We can all thank Bob Fischer for accurately describing it. Incidently, Bob publishes a newsletter called "Extensions" which features adaptations and enhancements to the Pro/File 2068 program. He has come up with some pretty neat things which you should check out. Extensions is a bargain at just $6. Write to Bob for your copy soon.

Another rather curious bug in the TS2068 32K Non-Volatile Ram board has been fixed. A very few of the first boards produced would end up with a dead battery after the board had been left in the computer for a few hours. We checked chips, batteries, diodes, virtually everything we could think of and still couldn't find the leak. Imagine our surprise when we found it was the battery prong touching the inside of the TS2068 case (which has a grounded conductive coating on it to reduce RF noise). It was a dead short! Anyway, a small piece of electricians tape over the battery prong makes the board last practically forever. We estimate that in "storage mode", that is, when it is removed from the computer and is powered solely by the on-board lithium cell, the memory will retain its contents for about 145 years!

Finally, here's a hopeful tidbit. Tom Bent, the creator of the 32K Non-volatile RAM, is hard at work figuring a way to use this memory on the TS1000! He reports that a TS1000 rear connector for the memory board is close to being operational.

ZX81 EXTENDED BASIC NOTES

Robert Shade from Philadelphia was studying the DEMO program which comes with Extended Basic and asked what function all the OUT commands perform. According to Frits Beniest, author of the program, and programmer extraordinaire, these commands were included to generate sound from the ZON-X sound generator. This peripheral is available in Europe, but to my knowledge, is not distributed here in North America.

If we had this device plugged in to the TS1000, we would have an autitory melee of bangs, roars, bells, and whistles accompanying the dazzling visual display. As it is, you could delete every OUT command, the only effect being a slightly faster program run. It is hard to imagine the demo going any faster though, even if it were written in machine code!